Regret-Based Decentralized Adaptation of Web Processes with Coordination Constraints

Yunzhou Wu *and* Prashant Doshi
LSDIS Lab, Dept. of Computer Science
University of Georgia
Athens, GA 30602
wuyzh@uga.edu, pdoshi@cs.uga.edu

Abstract

We present a new approach for adapting Web processes to input data volatility while respecting possible coordination constraints between the process participants. We focus on modeling the decision making process of service managers that are responsible for accomplishing the activities of an abstractly defined process. Compared to previous approaches that either employ a process manager overseeing all service managers or have a coordination mechanism which must be perfectly observed by all service managers, our approach does not need any central component but endows each service manager with the ability to communicate with other service managers. We define a regret-based coordination mechanism in order to motivate managers to respond to the communication. Our experiments demonstrate that our approach performs better than the previous decentralized approach while addressing the scalability problem suffered by the globally optimal centralized approach.

1 Introduction

Adapting to external events is a key requirement for dynamic and agile business processes. This is because the business environments in which the processes function seldom remain static over the lifecycle of the process. Atypical input and execution data, and new knowledge relevant to the process are some of the reasons necessitating adaptation. Some of the external events within this category, which we call *data volatility*, may be expected to occur with some chance, though usually it is low. For example, manufacturers routinely expect a small percentage of their orders to be delayed in arriving. If the shipment is delayed, the manufacturer may wait out the delay or its process may adapt by canceling the order and choosing a different supplier.

Realistic business processes often exhibit constraints between the disparate components of the processes. An example constraint is when the merchandise ordered at different points in the process must be compatible. Another example is that the date of the rental car and hotel reservation must be identical to the date of arrival at the airport. Hence, changing the service that provides goods (perhaps due to a delay in satisfying the order) in the first example or changing the date of the trip (due to unavailability of seats in an airplane) in the second example may need to be *coordinated* with changes at other points in the process. Adaptation of processes is further complicated in the presence of coordination constraints between services. Specifically, the state of all services participating in the constraints should be considered when deciding on a response to the external event.

We adopt the framework [10] that involves specifying *service managers* (SMs) that are responsible for accomplishing the activities of an abstractly defined process. While the SMs have local oversight over the service partners used for achieving their individual activities, a global *process manager* (PM) is also specified which has global oversight over the entire process. Within this framework, adaptation to external events expected with some chance is viewed as a decision-making problem, and the approaches are evaluated based on the cost-based optimality and computational efficiency of the adaptation. Verma et al. [10] describe two approaches for adaptation: (a) A centralized approach in which the PM deliberates and suggests the adaptive actions. This approach is cost-optimal but computationally expensive. (b) A decentralized approach in which the individual SMs recommend adaptative actions assisted by a naive coordination mechanism for meeting the constraints. Though suboptimal, the approach is efficient and scalable.

In this paper, we present a new approach for adapting processes to external events that are expected to occur with some chance while respecting possible coordination constraints between the process participants. Analogous to the decentralized approach of [10], the SMs are tasked with adaptation in response to relevant external events. Therefore, we do not assume the presence of a PM which may be difficult to implement for distributed processes. In contrast to the decentralized approach, all the SMs are not compelled to perform the coordinating action if any one of them decides to do so. Instead of a coordination mechanism whose state is perfectly observable to all the SMs, we endow each manager with the ability to communicate with the other managers, though there is a cost of communication. A manager may choose to first establish the global optimality of the coordinating action by soliciting the impact of the action on the other SMs. In order to motivate SMs to respond to the communication we define a *regret-based* mechanism. Specifically, other SMs may be able to influence the global decision in their favor by sending in their values of performing the coordinating action. Though exchanging messages with the other managers has a cost, a manager is motivated to meet the request as otherwise with some chance it may *regret* not doing so by being forced to perform a possibly suboptimal coordinating action.

We adopt a Web services (WSs) architecture, wherein we implement the abstract processes as WS-BPEL flows and the SMs are specified using WSDL files. Inter-activity dependencies such as coordination constraints are represented as shown in [9]. At runtime, when the process engine makes a call to a partner WS it is routed to the SMs. We empirically compare our approach to those described in [10] and evaluate its performance on the basis of the optimality of the adaptation and how efficiently it can be carried out. We show, using two problem domains, that our approach to adaptation results in a process that is more cost-effective in simulated volatile environments than the previous decentralized approach, while being computationally efficient.

2 Related Work

Much of the earlier work on adaptation concentrated on manually changing traditional processes at both the logic and instance levels, and automatically verifying the correctness of these changes. Both graph based techniques [7] and Petri-nets [3] have been used to evaluate the feasibility and correctness of changes in the control flow of running instances. In a similar vein, Aalst and Basten [8] proposed a Petri-net based theory for process inheritance which categorized the types of changes that do not affect other interacting processes. Muller et al. [5] used event-condition-action rules to make changes in running instances. None of these approaches have considered the issue of adaptation and the long term optimality of the adaptation, as we do in this research. Furthermore, these approaches are complementary because they focus on the correctness of the adaptations to the structure of the flows once the changes have been made. Notice that the rules of correctness may themselves change due to, for example, emergence of new knowledge relevant to the process. We seek to answer the question of how to adapt the processes to volatility as opposed to verifying the correctness of the change. We ensure correctness by representing the orderings and causal conflicts in the methods that generate the flows. Isolated attempts to address inter-activity dependencies in processes include [1] in which dependencies at the transactional level were enforced using scheduling. In this work, the focus was on generating feasible schedules without emphasis on being optimal.

In [2], several alternate plans are pre-specified at the logical, physical, and runtime levels. Depending on the type of changes in the environment, alternative plans from these three stages are selected. While capable of adapting to several different events, many of the alternative pre-specified plans may not be used making the approach inefficient, and there is no measure of the optimality of the resulting Web processes. A complementary line of work is [4], which adapts Web processes at run time to changes in the quality-of-service parameters of the service providers. This research investigates which service provider to query for revised information, and complements our work in this paper.

3 Example Applications

We give two example scenarios that may need adaptation while maintaining inter-activity dependencies.

Trip planning: We consider a Web process for organizing a trip that consists of booking an airline ticket, and hotel and rental car at the destination. We consider the event where a booking of the airline, hotel or rental car would subsequently need modification because the airline, hotel or rental car becomes unavailable, perhaps due to overbooking. In response, the trip planner may choose to change the car company, change the date of departure, change the destination airport to another one in the city, or simply wait in the hope that a vacancy may arise. We note that a change in the departure date will require coordinated re-bookings of all three: airline, hotel and the rental car. A change in the destination airport will need modification in the booking of the airline and the rental car agency. We illustrate the trip planning process and the coordination constraints in Fig. 1. The decision of changing the date is not trivial as substantial costs are incurred if a confirmed hotel or rental car reservation must be canceled and a new one booked. These costs may offset the benefit of changing the date and waiting may turn out to be a better option. The decision of changing the airport is similarly complex.

Supply chain for computer assembly: Another scenario we consider is the supply chain of a computer manufacturer (e.g. Dell) which operates on minimal inventory, and therefore incurs significant costs if its order is delayed. The computer manufacturer typically orders in bulk different com-

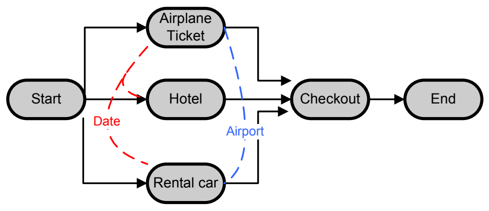


Figure 1. The trip planning process with coordination constraints. A change in the date of departure requires rebooking for all three activities, while a change in destination airport requires modifications to two activities.

puter parts from different suppliers. Since the parts must be assembled into a single computer, they must be compatible with each other. For example, the RAM must inter-operate with the motherboard. Therefore, if the delivery of the RAM is delayed and the manufacturer chooses to change the RAM supplier, the supplier of the motherboard must also be changed to preserve the compatibility constraint. [1]

As an example of the choice involved here, in deciding to change the RAM supplier the manufacturer must consider the consequences in terms of cost of ordering the motherboard from a new compatible supplier too. Of course, the cost of switching suppliers will vary with the state of the process. For example, if the delivery of the RAM is delayed and the motherboard has arrived, then a decision to change the RAM supplier would entail returning back the motherboard and changing the motherboard supplier. Such a decision might prove more costly than waiting out the delay in receiving the RAM. Additionally, the new supplier may also suffer a delay. The problem is to adapt optimally to the external events like delay while respecting the constraints.

4 Overview of Previous Approaches

We briefly review the framework and the centralized and decentralized approaches to adaptation that appear in [10].

4.1 Framework

The framework for the adaptive process consists of two main layers - the resources and the control managers. The resource layer consists of the resources in the process such as the process engine, the Web services (WSs), and the process configuration module. A corresponding manager controls each of the resources. Each process instance, for example, is under the control of a PM, which has global oversight and is responsible for supporting some of the adaptive properties of the process. Its duties include configuring the process with the help of the configuration module, listening or querying the various environment variables for changes and working with the SMs for adapting the process in response to external events while preserving the logical consistency. Each SM is associated with an activity in the process and is responsible for accomplishing it. To do this, SMs contain a precise description of the activity. A partner WS interacts with a SM rather than directly with the process instance.

Adapting to environmental volatility relevant to the process involves a choice: We may ignore the data or any new knowledge and carry out the process unchanged, or we may take adaptive measures in response to the events. Of course, ignorance begets an expected cost that may be too large to make it a good choice. On the other hand, if adapting to data volatility entails substantial and costly modifications to the process, performing them may be a suboptimal decision. Therefore, adaptation to external events expected to occur with some chance, is treated as a decision-making problem. Well studied ways of decision-making include stochastic frameworks such as Markov decision processes (MDPs) [6].

4.2 Markov Decision Processes (MDPs)

MDPs [6] are well known intuitive frameworks for sequential decision-making under uncertainty. In addition to modeling the uncertainty that pervades realistic environments, they provide a way to model costs and guarantee cost-based optimality of the decisions. An MDP is a tuple:

$$MDP = \langle S, PA, T, C, OC \rangle$$

where: S is the set of states of the process. $PA : S \to \mathcal{P}(A)$ is a function that gives the set of actions permissible from a state. Here, A is the set of possible actions and $\mathcal{P}(A)$ is its power set. $T : S \times A \times S \to [0, 1]$ is the *Markovian* transition function which models the probability of the resulting state on performing a permitted action from some local state ($Pr(s'|s, a)$). $C : S \times A \to \mathbb{R}$ is the cost function which gives the cost of performing an action in some state of the process. The parameter, OC, is the optimality criterion. In this paper, we minimize the expected cost over a finite number of steps, $n \in \mathbb{N}$. Additionally, each unit of cost incurred one step in the future is equivalent to $\gamma \in [0, 1]$ units at present.

We solve the MDP offline to obtain a *policy*. The policy is a prescription of the action that is cost-optimal given the state and the number of steps to go. Formally, a policy is, $\pi : S \times \mathbb{N} \to A$ where S and A are as defined previously, and $\mathbb{N}$ is the set of natural numbers. The advantage of a policy-based approach is that no matter what the state is, the policy will always prescribe the optimal action. In order to compute the policy, we associate each state with a value that represents the long term expected cost of performing

[1]While RAM from a new supplier might be compatible with the existing motherboard or the same supplier could provide another compatible RAM, we do not consider these alternatives to focus on the coordination constraint.

the optimal policy from that state. Let $V : S \times \mathbb{N} \rightarrow \mathbb{R}$ be the function that associates this value to each state. Then,

$$V^n(s) = \underset{a \in PA(s)}{min} \; Q^n(s,a) \tag{1}$$

$$Q^n(s,a) = C(s,a) + \gamma \sum_{s'} T(s'|s,a) V^{n-1}(s') \tag{2}$$

where Q^n is called the action-value function. Here, $n \in \mathbb{N}$ is the finite number of steps to be performed. Note that $\forall_{s \in S}, V^0(s) = 0$. The optimal action from each state is the one that optimizes the value function:

$$\pi^n(s) = \underset{a \in PA(s)}{argmin} \; Q^n(s,a) \tag{3}$$

4.3 Simple Approaches for Adaptation

The decision of how to react to external events becomes more difficult in the presence of inviolable constraints between the process activities. The PM having global knowledge of the entire process including the states of the SMs is capable of adapting optimally to the external events while satisfying the constraints. The M-MDP approach in [10] describes a *centralized* way of doing this that guarantees global optimality of the adaptation, but does not scale efficiently to large processes. Specifically, it groups together the state and action spaces of all the individual SMs, modeling it as a large joint decision problem.

One way to scale reasonably well to large processes with several activities is to allow each associated SM to make its own decisions on how to adapt to the external events. The MDP-CoM approach [10] formalizes each SM's behavior as an individual decision-making problem, and ensures coordination between SMs using a simple coordination mechanism (CoM). In this approach, when any of the SM signals its intent to perform a coordinating action, each SM must follow suit, no matter whether it's an optimal decision for the SM. However, this is precisely the source of the loss in optimality for the decentralized approach.

5 Decentralized Adaptation

As we mentioned in Section 4.3, approaches such as M-MDP and MDP-CoM [10] while offering ways of adapting the process to exogenous events without violating the coordination constraints, do so with significant limitations. We offer an approach for alleviating these difficulties (Fig. 2). To promote scalability, each SM models and solves it own decision-making problem, which will guide its interactions with the partner WSs. However, in order to handle the inter-activity coordination constraints, we design a more sophisticated CoM based on inter-SM communication, which attempts to establish the global optimality of a coordinating action before performing it. We do this without compelling any SM to go against its self-interests.

At a more technical level, we assume that all the SMs act at the same time step and actions of the other SMs are not observable. Each SM fully observes its local state and that of its CoM, and is cognizant of its possible participation in coordination constraints with some other SMs. Furthermore, the communication channels between the SMs are assumed to be reliable so that there is no data loss in the communication.

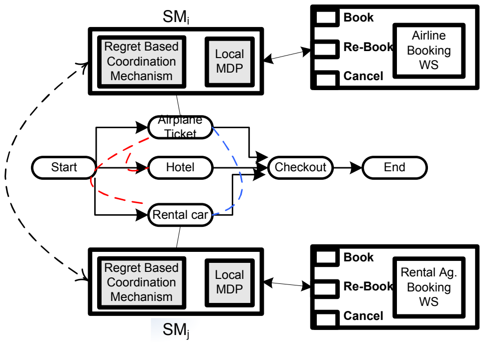


Figure 2. **SM k for hotel activity not shown for clarity. Each SM locally decides its action in response to events. SMs coordinate (dashed line denotes the coordination constraint) via communication (dashed arrow indicates communication channel) that is guided by local CoM.**

5.1 Formal Model

We model each SM's decision making process as a MDP. The MDP model for a SM, say i, is:

$$\mathcal{SM}_i = \langle S_i, PA_i, T_i, C_i, OC_i \rangle$$

where S_i, PA_i, T_i, C_i, and OC_i are as defined in Section 4.2. In this paper, we assume that each of the SMs optimizes w.r.t. a discounted finite number of lookahead steps, though in general they could have different optimality criteria. For our trip planning example, the MDP for SM i is:

Example 1. *An example local state of the SM i is* $\mathbf{B}\ \bar{\mathbf{A}}\ \bar{\mathbf{CD}}\ \bar{\mathbf{CA}}\ \bar{\mathbf{R}}$ *which denotes that i has booked a seat (*$\mathbf{B}$*) that has become unavailable (*$\bar{\mathbf{A}}$*), but it has not changed the date (*$\bar{\mathbf{CD}}$*) or the destination airport (*$\bar{\mathbf{CA}}$*) nor has it recieved the ticket (*$\bar{\mathbf{R}}$*). Possible actions for the SM i are: A_i = {* Book (B), Wait (W), ChangeAirline (CR), ChangeDate (CD), ChangeAirport (CA) *}. The action* Book *denotes the invocation of the relevant WS operations of the chosen airline to initiate a reservation,* Wait *is similar to a no operation (NOP),* ChangeAirline *denotes that a seat has been booked in a different airline company for the same itinerary,* ChangeDate *signifies the invocation of the* *relevant WS operation to change the date of the reservation and* ChangeAirport *signifies the invocation of the relevant WS operation to change the destination airport. The transition function is shown in Fig. 3(b). Due to lack of space, we do not show the cost function.*

5.2 Exogenous Events

In our example trip planning scenario, the planner must act in response to several events such as a notification of unavailability from the airline and a notification of receipt of the ticket. In order to ensure that the SM responds to these events optimally, they must be a part of our model. Since the events are external to the Web process, we label them as *exogenous*.

In order to model the exogenous events, we perform two steps as outlined in [10]: (1) We specify an expanded transition function for the SM i. In other words, $T_i^E : S_i \times A_i \times E_i \times S_i \rightarrow [0, 1]$, where E_i is the set of mutually exclusive events, and rest of the symbols were defined previously. For our example, E_i = {*Unavailable*, *Received*, None}. The expanded transition function models the uncertain effect of not only the SM's actions but also the exogenous events on the state space. We show the expanded transition function for the SM i in Fig. 3(a). (2) We define

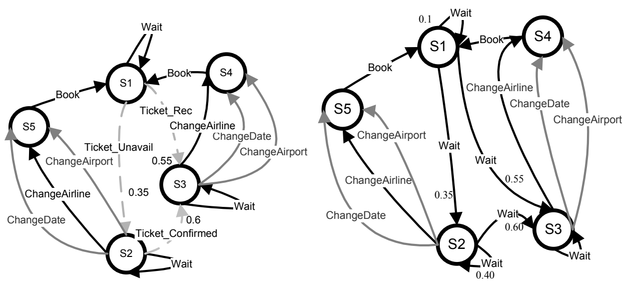


Figure 3. (a) **A simplified state transition diagram for SM i illustrating actions, events and probabilities. Transitions due to actions are depicted using solid lines and these are deterministic. Exogenous events are shown dashed. The numbers denote example probabilities of occurrence of events conditioned on the states.** (b) **The state transitions illustrating the transition function, T_i, for the SM i. Some transitions due to actions are now non-deterministic.**

a'priori a probability distribution over the occurrence of the exogenous events conditioned on the state of the SM. For example, let $Pr(\textit{Unavailable}|BA\bar{C}D\bar{C}A\bar{R}\,)$ = 0.35 be the probability that SM i's booking of airplane is unavailable.

We obtain the transition function, T_i, that is a part of the model defined in Section 5.1, by marginalizing or absorbing the events. Formally,

$$T_i(s'_i|s_i, a_i) = \sum_{e \in E_i} T_i^E(s'_i|s_i, a_i, e)Pr(e|s_i)$$

Here, T_i^E is obtained from step (1) and $Pr(e|s_i)$ is specified as part of the step (2) above. The marginalized transition function for the SM i is shown in Fig. 3(b).

6 Regret Based Coordination Mechanism

Decentralized approaches model each SM as self-interested that arrives at its own decision on how to best adapt to the exogenous events. While the decision-making is local, the presence of coordination constraints across multiple SMs may require that some actions be globally optimal for all the constrained SMs. As an example, if the SM that is booking an airplane decides to change the date of travel, then the SM booking the hotel and rental car must also follow suit. However, this may not be an optimal decision for the SM j, especially if it has already received its confirmation. The cost of canceling the reservation and booking a new one may outweigh the benefits of changing the date to the SM i. Consequently, mechanisms to ensure coordination between the SMs in order to preserve the coordination constraints are needed.

6.1 Formulation of R-CoM

If an SM, say i, guided by its local MDP decides to perform an action that needs to be coordinated, it first attempts to establish the global optimality of this action for all the SMs involved. This is because the coordinating action should be performed by all the SMs participating in the constraint. In order to do this, SM i broadcasts a request to all the other involved SMs soliciting the impact of performing the action on their local interactions with their partner WSs. Mathematically, in our example, SM i requests the other SMs j and k to send in the action-value for performing, say CD, $Q_j^n(s_j, CD)$ and $Q_k^n(s_k, CD)$, as well as W, $Q_j^n(s_j, W)$ and $Q_k^n(s_k, W)$, from their current states s_j and s_k, respectively, where Q_j^n and Q_k^n are their action-value functions defined previously in Eq. 2. Let $Q_i^n(s_i, CD)$ and $Q_i^n(s_i, W)$ be i's action-value for performing CD and W from state s_i, respectively. Then, SM i will decide to do the coordinating action, CD, if:

$$\textstyle\sum_{m=i,j,k} Q_m^n(s_m, CD) \leq \sum_{m=i,j,k} Q_m^n(s_m, W)$$

The above inequality formalizes the intuition that changing date is globally optimal.

In general, if there are M SMs, $m = 1, 2, ..., M$, then an SM that decides to perform the coordinating action, a_c, will carry it out if:

$$\sum_{m=1}^{M} Q_m^n(s_m, a_c) \leq \sum_{m=1}^{M} Q_m^n(s_m, \bar{a}_{c,m}) \qquad (4)$$

where $\bar{a}_{c,m}$ is the action, possibly other than a_c, that is locally optimal for SM m.

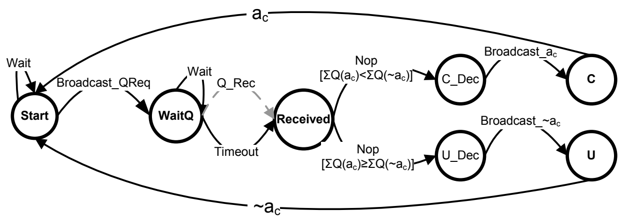


Figure 4. **The CoM that is followed when a SM decides to perform a coordinating action, a_c. The SM first solicits the impact of performing a_c on other SMs and then makes a decision based on the responses received. The decision is broadcasted to other SMs who must act it.**

We formalize this behavior of the SM using a *guarded* FSA shown in Fig. 4. The action, **Broadcast_QReq**, represents the solicitation of the action-values of the coordinating and non-coordinating actions, which may be locally optimal, from each of the other involved SMs. When the SM receives the action-values, denoted using an internal event, *Q_Rec*, the FSA transitions to the Received state. In this state, the inequalities (Eq. 4) represented as guards on the transitions are evaluated using the action-values sent back, and guide the transitions to either the Coordinate (C) or Uncoordinate (U) state at which the appropriate decision is broadcasted to the other SMs.

Perhaps, the more difficult challenge is to establish a motivation for the other SMs to respond to the broadcasted request, in particular because the other SM will incur a communication cost. In order to motivate why the other SM, say j, will communicate, we show that *with some chance it may regret not doing so*. Specifically, let a_c be the coordinating action, and $\bar{a}_{c,j}$ be some other action that is locally optimal for j. SM j will regret not communicating the action-values of its locally optimal non-coordinating action and the coordinating action at its current state, if the following conditions hold:

Condition 1:

$$\sum_{m=1}^{M-1} Q_m^n(s_m, a_c) + Q_j^n(s_j, a_c) \geq \sum_{m=1}^{M-1} Q_m^n(s_m, \bar{a}_c) + Q_j^n(s_j, \bar{a}_{c,j})$$

but SM j is compelled to perform the coordinating action, a_c, because $\sum_{m=1}^{M-1} Q_m^n(s_m, a_c) < \sum_{m=1}^{M-1} Q_m^n(s_m, \bar{a}_{c,m})$.

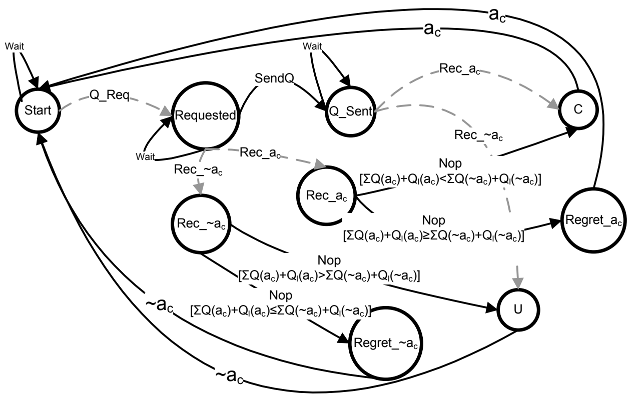
Alternatively, SM j's optimal action may be a_c, but it may regret not communicating if the following holds:

Condition 2:

$$\sum_{m=1}^{M-1} Q_m^n(s_m, a_c) + Q_j^n(s_j, a_c) < \sum_{m=1}^{M-1} Q_m^n(s_m, \bar{a}_c) + Q_j^n(s_j, \bar{a}_{c,j})$$

while $\sum_{m=1}^{M-1} Q_m^n(s_m, a_c) \geq \sum_{m=1}^{M-1} Q_m^n(s_m, \bar{a}_{c,m})$, thereby disallowing the coordinating action, a_c.

Intuitively, conditions 1 and 2 formalize the notion that SM j's response would have changed the globally optimal action selected by the initiating SM.



a_c
a_c
Wait
Wait
Start
Q_Req
Requested
SendQ
Q_Sent
Rec_a$_c$
C
Rec_~a$_c$
Wait
Rec_a$_c$
Rec_~a$_c$
Nop
[ΣQ(a$_c$)+Q$_l$(a$_c$)<ΣQ(~a$_c$)+Q$_l$(~a$_c$)]
Rec_a$_c$
Nop
[ΣQ(a$_c$)+Q$_l$(a$_c$)≥ΣQ(~a$_c$)+Q$_l$(~a$_c$)]
Regret_a$_c$
Rec_~a$_c$
Nop
[ΣQ(a$_c$)+Q$_l$(a$_c$)>ΣQ(~a$_c$)+Q$_l$(~a$_c$)]
U
~a$_c$
Nop
[ΣQ(a$_c$)+Q$_l$(a$_c$)≤ΣQ(~a$_c$)+Q$_l$(~a$_c$)]
Regret_~a$_c$
~a$_c$

Figure 5. **The CoM followed by a SM who receives a request for communicating the impact of performing a_c and $\bar{a}_c$ given its local state. If the SM decides not to respond to the request, it may be compelled to perform a locally sub-optimal action that could have been averted had it communicated. This results in the Regret_a_c or Regret_$\bar{a}_c$ states.**

We model this behavior of a SM on receiving a request for action-values using the guarded FSA shown in Fig. 5. Here, the nodes Regret_a_c and Regret_$\bar{a}_c$ denote the states where conditions 1 and 2 hold, respectively. **sendQ** is the action that denotes the response by the SM. In both the FSAs of Figs. 4 and 5, the internal events are marginalized into the automaton using the steps mentioned in Section 5.2. We define:

$$\text{Exp\_Regret}_j = [Pr(\langle \text{Regret\_a}_c, s_j\rangle | \mathbf{Nop}, \langle \text{Rec\_a}_c, s_j\rangle) + Pr(\langle \text{Regret\_}\bar{\text{a}}_c, s_j\rangle | \mathbf{Nop}, \langle \text{Rec\_}\bar{\text{a}}_c, s_j\rangle)] |Q_j^n(s_j, a_c) - Q_j^n(s_j, \bar{a}_{c,j})|$$

where s_j is the current state of j, $Pr(\langle \text{Regret\_a}_c, s_j\rangle \mid \mathbf{Nop}, \langle \text{Rec\_a}_c, s_j\rangle)$, $Pr(\langle \text{Regret\_}\bar{\text{a}}_c, s_j\rangle \mid \mathbf{Nop}, \langle \text{Rec\_}\bar{\text{a}}_c, s_j\rangle)$ together indicates the probability that SM j will regret its decision of not communicating its action-values. Exp_Regret$_j$ is thus the additional cost expected when the SM j is compelled to perform an action that is locally suboptimal. Then, by not communicating, SM j may incur the additional cost of Exp_Regret$_j$ with some chance. One way to arrive at the probabilities is to simulate the interaction between SMs. For example, we may check the number of times that SM j regrets given that it does not send action-values back upon arrival of **Q_Req** from SM i. We simulate this interaction a large number of times and decide the probabilities based on the simulated statistical results. Note that $Pr(\text{Regret\_}\bar{\text{a}}_c | \mathbf{Nop}, \text{Rec\_}\bar{\text{a}}_c))$ is computed analogously.

SM j responds to the request from i if its expected regret for not communicating is more or equal to the communication cost: Exp_Regret$_j$ $\geq$ Communication cost. While the FSAs are somewhat unwieldy, their structure is independent of the particular problem domain we study here. If multiple SMs decide to initiate a coordinating action we use semaphores or other mutual exclusion mechanisms.

6.2 Expanded Model and Solution

To ensure coordination the CoMs must be included in the SMs' decision-making processes. We do this by combining the MDP model defined previously in Section 5.1, with the CoM and call the new model, MDP-R-CoM. Within MDP-R-CoM, the state space is expanded to include the states of CoM as well: $\hat{S}_i = S_i \times Y$, Y is the state space of the CoM, Y = { Start, WaitQ, ReceivedQ, . . ., SentQ, Regret_a_c, Regret_$\bar{a}_c$, . . . }. The states are as in Figs. 4 and 5.

The action choices available to the SM are augmented with the communicative actions, $\hat{A}_i$= A_i $\cup$ { **Broadcast_QReq**, **Broadcast_**a_c, **Broadcast_**$\bar{a}_c$, **SendQ** }. To ensure that the SM, for example, performs the coordinating action iff the FSA is in the coordinate (C) or regret (Regret_a_c) state, we define the function, $\hat{PA}_i(\langle *, C\rangle) = a_c$, and remove the choice of a_c when the FSA is in the uncoordinated state, $\hat{PA}_i(\langle s_i, U\rangle) = PA_i(s_i)/a_c$. Here, '*' stands for a local state of SM i. We modify the PA_i function for the other actions analogously.

The transition function is the joint defined as: $\hat{T}_i : \hat{S}_i \times \hat{A}_i \times \hat{S}_i \rightarrow [0, 1]$. Here:

$$\begin{aligned}\hat{T}_i(\langle s_i', y'\rangle | a_i, \langle s_i, y\rangle) &= Pr(s_i'|y', a_i, \langle s_i, y\rangle)Pr(y'|a_i, \langle s_i, y\rangle)\\ &= T_i(s_i'|a_i, s_i)Pr(y'|a_i, \langle s_i, y\rangle)\end{aligned}$$

We illustrate the expanded transition function for a selected transition in the FSA:

$$\begin{aligned}&Pr(\langle \mathbf{BAC\bar{D}C\bar{A}R}, \textsf{Regret\_a}_\textsf{c}\rangle | \mathbf{Nop}, \langle \mathbf{BAC\bar{D}C\bar{A}R}, \textsf{Rec\_a}_\textsf{c}\rangle)\\ &= T_j(\mathbf{BAC\bar{D}C\bar{A}R}|\mathbf{Nop}, \mathbf{BAC\bar{D}C\bar{A}R})Pr(\textsf{Regret\_a}_\textsf{c}|\mathbf{Nop},\\ &\langle \mathbf{BAC\bar{D}C\bar{A}R}, \textsf{Rec\_a}_\textsf{c}\rangle)\\ &= 1 \times Pr(\textsf{Regret\_a}_\textsf{c}|\mathbf{Nop}, \langle \mathbf{BAC\bar{D}C\bar{A}R}, \textsf{Rec\_a}_\textsf{c}\rangle)\end{aligned}$$

The cost function, $\hat{C}_i : \hat{S}_i \times \hat{A}_i \rightarrow \mathbb{R}$, gives the cost of acting from the combined local state and the state of the CoM. The expanded cost function also specifies the cost of the communicative actions that are used to exchange information between the SMs. For our purposes, the state of the CoM does not matter in deciding the cost of the actions.

We associate with each local state of the SM and the state of the CoM, a value function that gives the expected cost of following an optimal policy from that state. Equation 2 forms the basis for computing the value function, while Eq. 3 computes the optimal policy for the MDP-R-CoM model, $\pi^i : \hat{S}_i \times \mathbb{N} \rightarrow A_i$. In these equations, we use the expanded model described in Section 6.2. Compared with MDP-CoM, our approach is truly decentralized because we do not need a central CoM that must be perfectly observed by all SMs. Furthermore, as we show in the next theorem our approach exhibits less or equivalent loss in optimality of the adaptation compared to MDP-CoM.

Theorem 1. *The expected cost of the adaptative actions generated by MDP-R-CoM is equivalent or less than those generated by MDP-CoM.*

Proof. Let $s_1, s_2, \ldots, s_M$ be the states of the SMs 1 to M, respectively. The theorem follows from showing that, for all possible states, $\sum_{m=1}^{M} V_m^{n,R-CoM}(s_m) \leq \sum_{m=1}^{M} V_m^{n,CoM}(s_m)$, where $V_m^{n,R-CoM}(\cdot)$ represents the expected long term cost (Eq. 1) of the SM$_m$ in the MDP-R-CoM approach, analagously for $V_m^{n,CoM}(\cdot)$, and their summation represents the global expected cost for the process. We consider two cases:

Case 1: No SM selects a coordinating action. Because in both the approaches each SM solves its own decision problem optimally in the absence of coordinating actions, $\sum_{m=1}^{M} V_m^{n,R-CoM}(s_m) = \sum_{m=1}^{N} V_m^{n,CoM}(s_m)$.

Case 2: One (or more) of the SM, say i, selects a coordinating action, a_c. Thus, $Q_n^i(S_i, a_c) < Q_n^i(S_i, \bar{a}_c)$. In MDP-CoM, the CoM will compel each SM to perform a_c in order to preserve the coordinating constraint. Thus the global expected cost will be: $\sum_{m=1}^{M} V_m^{n,CoM}(s_m) = \sum_{m=1}^{M} Q_m^n(s_m, a_c)$. In our MDP-R-CoM approach, the R-CoM will solicit the action-values of performing a_c and the best non-coordinating action to make the decision. Thus, $\sum_{m=1}^{M} V_m^{n,R-CoM}(s_m) =$ min { $\sum_{m=1}^{N} Q_m^n(s_m, a_c)$, $\sum_{m=1}^{N} Q_m^n(s_m, \bar{a}_{c,m})$ }. Thus, $\sum_{m=1}^{M} V_m^{n,R-CoM}(s_m) \leq \sum_{m=1}^{M} V_m^{n,CoM}(s_m)$. □

7 Model Elicitation from WS Descriptions

Given the partner WS descriptions in WSDL or OWL-S and service-level agreements between the participants, we show how the individual model parameters of SMs may be elicited from these descriptions. The actions are the WS operations that accomplish the activity. The probabilities of the events that make up the expanded transition function, T^E, may be common domain knowledge or learned from past experience. We assumed that the actions are deterministic, though our models may accommodate uncertain actions in a straightforward manner. The cost of using WSs may be obtained from the serviceParameter section of the OWL-S description or from the SLA-parameter section of the WSLA document between the service users and providers.

8 Experiments

We empirically evaluate the performance of MDP-R-CoM using the two problems, trip planning and supply

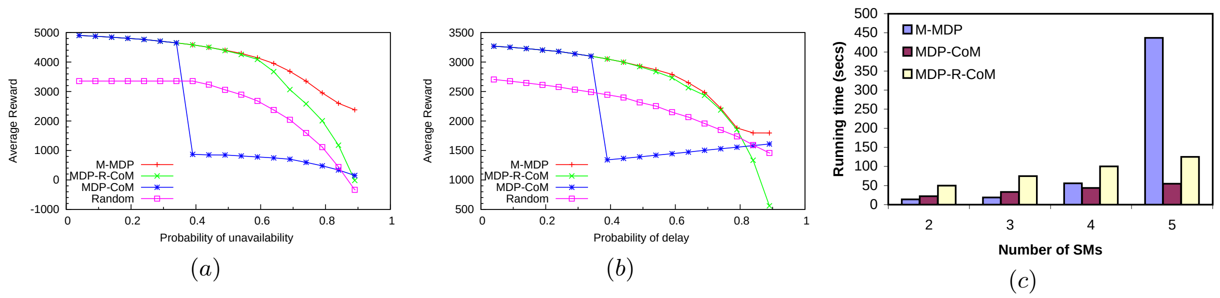


Figure 6. **Performances of the adaptation approaches for the two problem domains, (a) trip plan, and (b) supply chain. Notice that the MDP-R-CoM approach performs better than MDP-CoM though worse than the globally optimal M-MDP approach. (c) Execution times of the adaptation approaches (Xeon 3.0GHz, 2 GB RAM, Linux). Running time of MDP-R-CoM scales well to multiple SMs.**

chain, and compare it with the previous approaches of M-MDP and MDP-CoM [10] and a random selection approach. We show the performances of the approaches in Fig. 6. Each data point in the plots is the average of 3,000 runs in a simulated environment where the events, *unavailability* (for trip planning) and *delay* (for supply chain), were sampled according to distributions that were varied. Each run consisted of the SMs performing 20 steps. We added the total rewards obtained by each of the SMs; specifically a SM incurred a cost for waiting if the ticket was unavailable or the order was delayed and obtained a reward in the state where either were received.

As we may expect, MDP-R-CoM performed better than both, the approach of MDP-CoM and random selection. We observed that in many cases, the SMs had an expected regret that was large enough motivating them to respond to solicitations. Thus, often the SM originating the coordinating action made decisions that were globally optimal. This is primarily responsible for the better performance of MDP-R-CoM. The SMs in the three approaches that used decision-making all opted to wait if the probability of unavailability or delay was small, which explains why they perform identically initially. This is because the SMs believe that it is more likely that they will receive the ticket (or order) once it is unavailable (or delayed). However, as the probability of unavailability further increases, the SMs choose to either change the date or the airport based on which action is optimal as decided by the CoM. Notice the dip in the performance of MDP-CoM; this is because the SMs decide to change the date or airport rather than wait. As the CoM in this approach compels all SMs to do the coordinating action if any one of them decides to do so without regard to their local states, in many runs, confirmed reservations had to be canceled and received orders had to be returned.

The run times of the adaptive process is shown in Fig. 6. Because each SM performs its own decision-making, the MDP-R-CoM scales reasonably well to multiple SMs analogous to MDP-CoM. However, because of a more sophisticated CoM that utilizes communication and internal events, its run time is more than MDP-CoM.

In summary, the MDP-R-CoM approach performs decisions that are often globally optimal though the individual SMs are modeled as self-interested. In addition, its decentralized nature allows it to scale well to multiple SMs though its running time is more than that of MDP-CoM because of its more sophisticated CoM.

References

[1] P. C. Attie, M. P. Singh, A. P. Sheth, and M. Rusinkiewicz. Specifying and enforcing intertask dependencies. In *VLDB*, pages 133–145, 1993.

[2] G. Chafle, K. Dasgupta, A. Kumar, S. Mittal, and B. Srivastava. Adaptation in web service composition and execution. In *ICWS, Industry Track*, 2006.

[3] C. Ellis, K. Keddara, and G. Rozonberg. Dynamic change within workflow systems. In *COOCS*, pages 10–21, 1995.

[4] J. Harney and P. Doshi. Adaptive web processes using value of changed information. In *ICSOC*, pages 179–190, 2006.

[5] R. Muller, U. Greiner, and E. Rahm. Agentwork: a workflow system supporting rule-based workflow adaptation. *J. of Data and Knowledge Engg.*, 51(2):223–256, 2004.

[6] M. L. Puterman. *Markov Decision Processes:Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 1994.

[7] M. Reichert and P. Dadam. Adeptflex-supporting dynamic changes of workflows without losing control. *J. of Intelligent IS*, 10(2):93–129, 1998.

[8] W. van der Aalst and T. Basten. Inheritance of workflows: an approach to tackling problems related to change. *Theoret. Comp. Sc.*, 270(1-2):125–203, 2002.

[9] K. Verma, R. Akkiraju, R. Goodwin, P. Doshi, and J. Lee. On accommodating inter service dependencies in web process flow composition. In *AAAI Spr. Symp.*, pages 37–43, 2004.

[10] K. Verma, P. Doshi, K. Gomadam, J. Miller, and A. Sheth. Optimal adaptation in web processes with coordination constraints. In *ICWS*, pages 257–264, 2006.